Shrinking alternating two-pushdown automata (New Aspects of Theoretical Computer Science)

AUTHOR(S):

Otto, Friedrich; Moriya, Etsuro

数理解析研究所講究録 1325 巻 2003 年 191-196

Shrinking alternating two-pushdown automata

ドイツ連邦共和国・カッセル大学　**F. オットー (Friedrich Otto)**
Fachbereich Mathematik/Informatik, Universität Kassel
34109 Kassel, Germany

and

早稲田大学・教育学部　**守屋悦朗 (Etsuro Moriya)***
Department of Mathematics, School of Education
Waseda University, Shinjuku-ku, Tokyo, 169-8050, Japan

Abstract

The alternating variant of the shrinking two-pushdown automaton of Buntrock and Otto (1998) is introduced. It is shown that the class of languages accepted by these automata is contained in the class of deterministic context-sensitive languages, and that it contains a PSPACE-complete language. Hence, the closure of this class of languages under log-space reductions coincides with the complexity class PSPACE.

1 Introduction

In 1986 Dahlhaus and Warmuth published their paper [5] on the class GCSL of *growing context-sensitive languages*, in which they proved that the membership problem for a language from this class is decidable in polynomial time. Here a language is called *growing context-sensitive*, if it is generated by a phrase-structure grammar containing only rules that are strictly monotonous, that is, for each rule of the grammar the left-hand side is strictly shorter than the right-hand side.

Since then this language class has received a lot of attention. First Buntrock and Loryś have shown that GCSL is an abstract family of languages in the sense of Ginsburg, Greibach and Hopcroft [6], and that it can be characterized by less restricted classes of grammars [1]. Then Buntrock and Otto [2] have shown that GCSL is the class of languages that are accepted by the so-called *shrinking two-pushdown automata* (sTPDA). These two-pushdown automata are shrinking with respect to a weight function that assigns a positive weight to each pushdown symbol and to each internal state, which gives a weight for each configuration by adding up the weights of the contents of the two pushdown stores and of the actual state. It is required that in each transition of the automaton the weight of the actual configuration decreases (see Section 2 for the definition). Interestingly the deterministic variant of the sTPDA characterizes the class of *Church-Rosser languages* CRL [2, 15], which were defined by McNaughton et al. using length-reducing and confluent string-rewriting systems [13, 14]. In fact McNaughton asks in [12] whether GCSL should be inserted in the Chomsky hierarchy as the class of "type one-and-a-half languages."

For the class CRL further characterizations by deterministic automata have been obtained. Motivated by the "analysis by reduction" used in linguistics, Jančar, Mráz, Plátek, and Vogel have defined and investigated the *restarting automaton* RRWW and many variants thereof [8, 9, 10]. An RRWW-automaton has a finite control and a read/write-head with a finite look-ahead working on a list of symbols. It can perform three kinds of operations: a *move-right step*, which shifts the read/write-window one position to the right and possibly changes the actual state, a *rewrite step*, which replaces the contents of the read/write-window by a shorter string, moving the head to the position immediately to the right of the newly written string and possibly changing the actual state, and a *restart step*, which moves the read-/write-head back to the left end of the list, and puts the automaton back into its inital state. Such an automaton works in cycles, and it is required that it performs exactly one rewrite step in each cycle. It should be stressed that during a rewrite step the length of the list actually decreases.

Apart from the basic model various restricted types of RRWW-automata have been considered, among them the RWW-automata, various monotonous versions and the deterministic variants. It has been observed that the class CFL of context-free languages coincides with the class of languages that are accepted by the so-called *monotonous*

RWW- and RRWW-automata [10], and that the deterministic RWW- and RRWW-automata yield other characterizations of the class CRL [16, 18]. Interestingly, these characterizations do not carry over to the corresponding nondeterministic classes [11, 17]: each growing context-sensitive language is accepted by some R(R)WW-automaton, but there is an R(R)WW-automaton that accepts the *Gladkij language* $L_{\mathrm{Gl}} := \{w\#w^R\#w \mid w \in \{a,b\}^*\}$, which is not growing context-sensitive [3]. Only by restricting attention to *weakly monotonous* R-(R)RW-automata we obtain the class GCSL [11]. Thus, while the sDTPDA is as powerful as the deterministic R(R)WW-automaton, the sTPDA is less powerful than the unrestricted R(R)WW-automaton. This raises the question of whether there is a more powerful variant of the sTPDA that is equivalent in computational power to the unrestricted R(R)WW-automaton.

As alternation is a powerful generalization of nondeterminism, we define and investigate here the alternating variant of the shrinking TPDA, the *shrinking alternating* TPDA (sATPDA). We will show that the increase in computational power obtained by alternation is not as big as one might expect, as the sATPDA only accepts languages that are deterministic context-sensitive. This is done by providing a simulation of an sATPDA by a deterministic linear space-bounded Turing machine. On the other hand, we will see that a suitably encoded version of the language QBF of quantified Boolean formulas, which is known to be PSPACE-complete, is accepted by some sATPDA, which implies that the closure of the class $\mathcal{L}$(sATPDA) under log-space reductions yields the complexity class PSPACE. Also we will see that the language class $\mathcal{L}$(sATPDA) is an abstract family of languages that in addition is closed under complementation, and therewith under intersection, and reversal.

As the closure of $\mathcal{L}$(RWW) and $\mathcal{L}$(RRWW) under log-space reductions yields the complexity class NP [11, 17], this gives some indication that the sATPDA is even more powerful than the R(R)WW-automaton. Unfortunately the exact relationship between the sATPDA and the R(R)WW-automata remains open.

2 Definitions and notation

Here we introduce some basic notation. Then we give the definition of the shrinking TPDA and its alternating variant, the sATPDA, and we describe a particular sATPDA for the Gladkij language L_{Gl}.

Let Σ be a finite alphabet. Then Σ^* denotes the set of strings over Σ including the empty string ε, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The following definition is taken from [2].

Definition 2.1. *A* two-pushdown automaton (TPDA) *is a nondeterministic automaton with two pushdown stores. Formally, it is defined as a 7-tuple* $M = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$, *where*

- Q *is the finite set of states,*
- Σ *is the finite input alphabet,*
- Γ *is the finite tape alphabet with* $\Gamma \supsetneq \Sigma$ *and* $\Gamma \cap Q = \emptyset$,
- $q_0 \in Q$ *is the initial state,*
- $\bot \in \Gamma \setminus \Sigma$ *is the bottom marker of the pushdown stores,*
- $F \subseteq Q$ *is the set of accepting states, and*
- $\delta\colon Q \times \Gamma \times \Gamma \to \mathcal{P}_{fin}(Q \times \Gamma^* \times \Gamma^*)$ *is the transition relation. Here* $\mathcal{P}_{fin}(Q \times \Gamma^* \times \Gamma^*)$ *denotes the set of finite subsets of* $Q \times \Gamma^* \times \Gamma^*$.

M *is a* deterministic two-pushdown automaton (DTPDA), *if* δ *is a (partial) function from* $Q \times \Gamma \times \Gamma$ *into* $Q \times \Gamma^* \times \Gamma^*$.

A configuration of a TPDA M is described as uqv, where $q \in Q$ is the actual state, $u \in \Gamma^*$ is the contents of the first pushdown store with the first letter of u at the bottom and the last letter of u at the top, and $v \in \Gamma^*$ is the contents of the second pushdown store with the last letter of v at the bottom and the first letter of v at the top. For an input string $w \in \Sigma^*$, the corresponding initial configuration is $\bot q_0 w \bot$. The TPDA M induces a computation relation $\vdash_M^*$ on the set of configurations (see, e.g., [7]), which is the reflexive, transitive closure of the single-step computation relation $\vdash_M$. The TPDA M accepts with its second pushdown store empty, that is, the language $L(M)$ accepted by M is defined as

$$L(M) := \{w \in \Sigma^* \mid \bot q_0 w \bot \vdash_M^* \alpha q \text{ for some } q \in F \text{ and } \alpha \in \Gamma^*\}.$$

In general the TPDA is as powerful as the Turing machine. However, we are only interested in certain restricted TPDA's.

A *weight function* for a TPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, \bot, F)$ is a mapping g that assigns a positive integer as weight to each symbol $a \in Q \cup \Gamma$. This mapping is extended to a morphism from $(Q \cup \Gamma)^*$ to $(\mathbb{N}, +)$ in the obvious way. To simplify the notation this morphism will also be denoted by g.

Definition 2.2. *A* (D)TPDA M *is called* shrinking *if there is a weight function* g *for* M *such that, for all* $q \in Q$ *and* $u, v \in \Gamma$, $(p, u', v') \in \delta(q, u, v)$ *implies that* $g(u'pv') < g(uqv)$. *By* s(D)TPDA *we denote the corresponding class of shrinking* (D)-TPDA*s*.

Without loss of generality we can assume for an sTPDA M that the bottom marker $\perp$ can only occur at the bottom of a pushdown store and that no other symbol can occur at that place, and that M halts with its right-hand pushdown store empty and the left-hand one containing at most the symbol $\perp$.

Proposition 2.3. (a) *A language is accepted by an* sTPDA *if and only if it is a growing context-sensitive language* [2].

(b) *A language is accepted by an* sDTPDA *if and only if it is a Church-Rosser language* [2, 15].

Finally, we are prepared to introduce the main topic of this note, the alternating shrinking TPDA.

Definition 2.4. *An* alternating TPDA $M = (Q, U, \Sigma, \Gamma, \delta, q_0, \perp, F)$ *is a* TPDA $(Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$ *for which a subset* U *of the set of states* Q *is designated as the set of* universal states. *The states in the difference set* $Q \setminus U$ *are accordingly called* existential states. *An alternating* TPDA (ATPDA) *is called* shrinking, *if the underlying* TPDA *is shrinking. By* sATPDA *we denote the class of all shrinking* ATPDA*s*.

We need to define the language accepted by an ATPDA. For doing so we introduce the notion of a *successful configuration* of an ATPDA.

Let $M = (Q, U, \Sigma, \Gamma, \delta, q_0, \perp, F)$ be an ATPDA, and let uqv be a configuration of M, where $q \in Q$ and $u, v \in \Gamma^*$. If $v = \varepsilon$ and $q \in F$, then uqv is a successful configuration. If q is an existential state, and there is a transition $uqv \vdash_M xpy$ such that xpy is successful, then also uqv is a successful configuration. Finally, assume that q is a universal state, and let $x_1p_1y_1, x_2p_2y_2, \ldots, x_kp_ky_k$ be the finitely many configurations that M can reach in a single step from uqv. Then uqv is a successful configuration, if all the configurations $x_ip_iy_i$, $1 \leq i \leq k$, are successful. Now the language $L(M)$ accepted by the ATPDA M is defined as follows:

$$L(M) := \{\, w \in \Sigma^* \mid \perp q_0 w \perp \text{ is a successful configuration of } M \,\}.$$

We are interested in the class $\mathcal{L}$(sATPDA) of languages that are accepted by sATPDAs. As each sTPDA is an sATPDA without universal states, we see that GCSL $\subseteq \mathcal{L}$(sATPDA) holds. In the following example we present an sATPDA that accepts the Gladkij language L_{Gl}.

Example 2.5. Let $M_{\mathrm{Gl}} := (Q, U, \Sigma, \Gamma, \delta, q_0, \perp, F)$ be defined by taking $Q := \{q_0, q_1, \ldots, q_6\}$, $U := \{q_1\}$, $\Sigma := \{a, b, \#\}$, $\Gamma := \Sigma \cup \{a_1, b_1, \perp\}$, $F := \{q_6\}$, and let δ be given through the following list of transition steps:

$q_0a \to a_1q_0$	$q_2b \to b_1q_2$	$b_1q_4b \to q_4$
$q_0b \to b_1q_0$	$q_2\# \to q_3$	$\perp q_4\# \to \perp q_5$
$q_0\# \to q_1\#$	$a_1q_3a \to q_3$	$q_5a \to q_5$
$q_1\# \to \#q_2$	$b_1q_3b \to q_3$	$q_5b \to q_5$
$q_1\# \to q_4$	$\#q_3\perp \to \#q_6$	$q_5\perp \to q_6$
$q_2a \to a_1q_2$	$a_1q_4a \to q_4$	

Further, we define a weight function $g : Q \cup \Gamma \to \mathbb{N}_+$ by taking

$$g(a) := g(b) := 2, \quad g(a_1) := g(b_1) := 1,$$
$$g(\#) := g(\perp) := 1, \quad g(q_i) := 7 - i \ (i = 0, \ldots, 6).$$

It is easily verified that the ATPDA M_{Gl} is shrinking with respect to this weight function. It remains to verify that $L(M_{\mathrm{Gl}}) = L_{\mathrm{Gl}}$ holds.

Let $x \in \Sigma^*$ be a given input. If x contains less than two or more than two occurrences of the symbol $\#$, then it is easily seen that M_{Gl} will not accept this input. Thus, we may assume that the input is of the form $x = u\#v\#w$ for some strings $u, v, w \in \{a, b\}^*$. Hence, the initial configuration is $\perp q_0u\#v\#w\perp$. Starting from this configuration M_{Gl} will reach the configuration $\perp u_1q_1\#v\#w\perp$, where u_1 denotes the image of u under the natural isomorphism from $\{a, b\}^*$ onto $\{a_1, b_1\}^*$. The state q_1 is the only universal state of M_{Gl}. The configuration $\perp u_1q_1\#v\#w\perp$ has exactly two immediate successor configurations:

(a) $\perp u_1\#q_2v\#w\perp$ and (b) $\perp u_1q_4v\#w\perp$.

Starting from (a) an accepting configuration is reached if and only if $w = v^R$, and starting from (b) an accepting configuration is reached if and only if $u = v^R$. Thus, these two computations lead both to acceptance if and only if $u = v^R = w$ holds. Hence, M_{Gl} does indeed accept the language L_{Gl}. □

As the language L_{Gl} is not growing context-sensitive, we obtain the following proper inclusion, which shows that the sATPDA is strictly more powerful than the sTPDA.

Corollary 2.6. GCSL $\subsetneq \mathcal{L}$(sATPDA).

3 An upper bound

Here we establish an upper bound for the computational power of the sATPDA by showing the following result.

Theorem 3.1. $\mathcal{L}(\text{sATPDA}) \subseteq \text{DSPACE}(n)$, *that is, if a language is accepted by some* sATPDA, *then it is deterministic context-sensitive.*

Proof. Let $M = (Q, U, \Sigma, \Gamma, \delta, q_0, \bot, F)$ be an sATPDA with weight function g, and let $\alpha := \max\{\, g(a) \mid a \in Q \cup \Gamma \,\}$. Obviously, we can restrict our attention to non-empty inputs. For each input $w \in \Sigma^+$, each computation of M on input w contains at most

$$g(\bot q_0 w \bot) \leq \alpha \cdot (|w| + 3) \leq 4\alpha \cdot |w|$$

many steps. Thus, the behaviour of the sATPDA M on an input $w \in \Sigma^+$ can be described by a computation tree of depth at most n, where $n := 4\alpha \cdot |w|$. As in the proof of Theorem 3.2 of [4] a deterministic Turing machine T can traverse this computation tree calculating the result of M's computation on input w. At any point in its computation T needs to store only the actual node of the computation tree of M it is currently visiting and the position of this node in the computation tree. As each configuration that M can reach during its computation on w is linearly bounded in the length of w, and as the depth of the computation tree is bounded by the number n, it follows that T can be realized in such a way that it is linear space-bounded. Hence, the language $L(M)$ is indeed deterministic context-sensitive. □

Thus, while alternation increases the expressive power of the pushdown automaton from the class CFL to the complexity class $\bigcup_{c>0} \text{DTIME}(c^n)$ [4], it increases the power of the sTPDA, which is stricly more powerful than the pushdown automaton, just to deterministic linear space.

4 A lower bound

Next we show that the sATPDA accepts the PSPACE-complete language L_{QBF}, thus providing a lower bound for its computational power.

Let $V := \{\, v_i \mid i \geq 0 \,\}$ be a set of Boolean variables, and let $\Sigma_0 := \{\exists, \forall, \neg, \wedge, \vee, (,)\}$, where $\neg$ is the symbol for *negation*, $\wedge$ denotes *conjunction*, and $\vee$ denotes *disjunction*. The set of *Boolean formulas* over V is defined inductively as usual: for each $v \in V$, v is a Boolean formula, and if F_1 and F_2 are Boolean formulas, then so are $(F_1 \vee F_2)$, $(F_1 \wedge F_2)$, and $(\neg F_1)$. A *quantified Boolean formula* is an expression of the form

$$Q_1 v_1 Q_2 v_2 \cdots Q_k v_k F,$$

where each Q_i is the existential quantifier $\exists$ or the universal quantifier $\forall$, and F is a Boolean formula containing (at most) the variables $v_1, \ldots, v_k$.

Under the standard logical interpretation each quantified Boolean formula yields the truth value *true* or *false*. By QBF we denote the language of all quantified Boolean formulas that yield the truth value *true*.

Let $\Sigma := \Sigma_0 \cup \{x, a\}$, and let $c : V \to \{x, a\}^*$ denote the unary encoding $v_i \mapsto x^2 a^i$ $(i \geq 0)$. For $\alpha \in (V \cup \Sigma_0)^*$, $c(\alpha)$ denotes the string from Σ^* that is obtained from α by replacing each variable occurrence v_i by its unary encoding $c(v_i)$. The language L_{QBF} is then defined as

$$L_{\text{QBF}} := \{\, c(\alpha) \mid \alpha \in \text{QBF} \,\}.$$

Example 4.1. The formula $\forall v_1 \exists v_2 \forall v_3\, (v_1 \vee \neg v_2) \wedge (v_2 \vee (\neg v_1 \wedge v_3) \vee \neg v_3)$ belongs to the set QBF, as is easily verified. Hence, the language L_{QBF} contains the string $\forall x^2 a \exists x^2 a^2 \forall x^2 a^3\, (x^2 a \vee \neg x^2 a^2) \wedge (x^2 a^2 \vee (\neg x^2 a \wedge x^2 a^3) \vee \neg x^2 a^3)$.

From the literature on complexity theory it is well-known that the language L_{QBF} is PSPACE-complete under log-space reductions, see, e.g., [20]. Here we establish the following result about this language.

Theorem 4.2. *There is an* sATPDA *that accepts the language* L_{QBF}.

Proof. We describe an sATPDA M for the language L_{QBF}. The sATPDA M will work in several phases as follows:

1. It checks that the input is of the form
$$Q_1 c(v_{i_1}) \cdots Q_k c(v_{i_k})\, c(F),$$
where $v_{i_1}, \ldots, v_{i_k} \in V$ and $F \in (V \cup \Sigma_0)^*$.
2. It verifies that all the variables $v_{i_1}, \ldots, v_{i_k}$ are pairwise distinct.
3. It checks that the string F only contains occurrences of these variables.
4. It guesses truth values for all variable occurrences.
5. For each $j \in \{1, \ldots, k\}$, it verifies that for each occurrence of the variable v_{i_j} the same value has been chosen.
6. It checks whether F is a correct Boolean formula while evaluating F under the chosen assignment.

Below we describe in short how the sATPDA M does all this. We consider each of the above phases in turn.

Phase 1. This amounts to checking that the given input belongs to the regular language $(\{\exists,\forall\}\cdot x^2\cdot a^*)^+\cdot(\{\neg,\wedge,\vee,(,)\}\cup x^2\cdot a^*)^+$, which can simply be done by moving the input from the right-hand pushdown store onto the left-hand pushdown store and back again, using two disjoint copies of the input alphabet in order to realize this computation in a weight-decreasing manner. If the given input does not belong to this regular language, then M rejects it, otherwise the input w is of the form $Q_1x^2a^{i_1}\cdots Q_kx^2a^{i_k}F$ for some string $F\in(\{\neg,\wedge,\vee,(,)\}\cup x^2\cdot a^*)^+$. In the latter case M enters a universal state q_{U_1}, for which there are exactly three alternatives of how to proceed, one for each of the Phases 2 to 4.

Phase 2. Universally M chooses two indices r and s satisfying $1\le r<s\le k$, and it compares i_r to i_s. If i_r and i_s coincide, then M halts and rejects, otherwise it accepts.

As we use universal states, we can guarantee that all different choices of r and s are considered. Thus, this phase does not lead to rejection only if all the variables in the quantifier prefix of the given input are pairwise distinct.

Phase 3. Universally M chooses a variable occurrence x^2a^j within the string F, and then it existentially chooses an index $r\in\{1,\dots,k\}$. It now compares the numbers i_r and j. If i_r and j do not coincide, then M halts and rejects, otherwise it accepts.

As we use universal states, we can guarantee that all different choices of variable occurrences in F are considered. Thus, this phase leads to acceptance only if all variables occurring in F are listed in the quantifier prefix.

Phase 4. For each variable in the quantifier prefix a truth value t_j, $1\le j\le k$, is chosen, where this choice is realized using a universal state if the corresponding variable is universally quantified, and otherwise it is realized using an existential state.

Next also each variable occurrence in the string F is associated with a truth value. This is done by pushing F from the right-hand pushdown store onto the left-hand pushdown store and back, while replacing each variable occurrence x^2a^j by xtb^j for some truth value $t\in\{0,1\}$. As in Phase 1 additional copies of the input alphabet are used in order to realize this computation in a weight-decreasing manner.

Then M enters a universal state q_{U_2}, for which there are exactly two alternatives of how to proceed, one for each of the last two phases.

Phase 5. Using the same strategy as in Phase 3 it is checked that each variable occurrence in the string F has been assigned the same truth value as the corresponding variable in the quantifier prefix. This phase accepts only if the assignment is correct.

Phase 6. Finally M verifies that the string F is in fact (the encoding of) a Boolean formula in the sense of the above definition, and that it evaluates to the truth value *true* under the actual assignment chosen in Phase 4. For doing so the quantifier prefix is deleted, and in the string F each variable occurrence of the form xtb^j is simply replaced by the truth value t. Then by simulating a deterministic pushdown automaton M checks whether F is of the correct syntactical form, and it evaluates F under the given replacement of variables by truth values. It accepts if and only if F is of the correct syntactical form and if it evaluates to *true*.

From this description we see that the sATPDA M satisfies $L(M)=L_{\mathrm{QBF}}$, which proves the theorem. □

Together with Theorem 3.1 this theorem yields the following.

Corollary 4.3. $\mathsf{LOG}\cdot\mathcal{L}(\mathsf{sATPDA})=\mathsf{PSPACE}$.

Recall from [2, 5] that $\mathsf{LOG}\cdot\mathsf{CFL}=\mathsf{LOG}\cdot\mathsf{GCSL}=\mathsf{LOG}\cdot\mathcal{L}(\mathsf{sTPDA})\subseteq\mathrm{P}$ holds, that is, with respect to the closure under log-space reductions the use of alternation increases the power of the sTPDA from the complexity class LOG·CFL to the complexity class PSPACE.

5 Closure properties

Finally we turn to the closure properties of the language class $\mathcal{L}(\mathsf{sATPDA})$.

Theorem 5.1. *The language class $\mathcal{L}(\mathsf{sATPDA})$ is closed under union, intersection, and complementation, reversal, product and iteration, ε-free morphisms, and inverse morphisms.*

All these closure properties can be shown by constructing corresponding sATPDAs [19]. However, $\mathcal{L}(\mathsf{sATPDA})$ is not closed under arbitrary morphisms, as already its subclass GCSL is a basis for the recursively enumerable languages [3].

It remains the question of whether the language class $\mathcal{L}$(sATPDA) can be characterized as the closure of the class GCSL under certain language-theoretical operations. The class GCSL is not closed under intersection or complementation [3], but as it is contained in the complexity class P, we see that the Boolean closure $\mathcal{B}$(GCSL) of GCSL, that is, the closure of GCSL under intersection, union and complementation, is still contained in P. Thus, the closure of $\mathcal{B}$(GCSL) under log-space reductions is contained in P, and so $\mathcal{B}$(GCSL) = $\mathcal{L}$(sATPDA) would imply by Corollary 4.3 that P and PSPACE coincide.

6 Concluding remarks

In [17] it has been observed that there is an RRWW-automaton that accepts the NP-complete language L_{3SAT}. Further, this result has been improved in [11] by showing that there is already an RWW-automaton accepting a variant of this language. In summary, we obtain the following sequence of inclusions, where DR(R)WW denotes the deterministic R(R)WW-automata:

$$\begin{array}{lllll} \mathrm{CRL} = \mathcal{L}(\mathrm{sDTPDA}) = & \mathcal{L}(\mathrm{DR(R)WW}) & \subset & \mathcal{L}(\mathrm{sTPDA}) \\ = \mathrm{GCSL} & \subset \mathrm{LOG{\cdot}CFL} & = & \mathrm{LOG{\cdot}GCSL} \\ \subseteq \mathrm{P} & \subseteq \mathrm{NP} & = & \mathrm{LOG \cdot R(R)WW} \\ \subseteq \mathrm{PSPACE} & = \mathrm{LOG} \cdot \mathcal{L}(\mathrm{sATPDA}). & & \end{array}$$

It remains to compare $\mathcal{L}$(sATPDA) to $\mathcal{L}$(RWW) and to $\mathcal{L}$(RRWW). They all contain the class GCSL properly, and they are all contained in CSL, but what is the exact relationship between them?

References

[1] G. Buntrock, K. Loryś. On growing context-sensitive languages. In W. Kuich, ed., *Proc. of 19th ICALP*, Lect. Notes Comput. Sci. 623, pp. 77–88. Springer, Berlin, 1992.

[2] G. Buntrock, F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Inform. Comput.*, 141:1–36, 1998.

[3] G. Buntrock. *Wachsende kontext-sensitive Sprachen.* Habilitationsschrift, Fakultät für Mathematik und Informatik, Universität Würzburg, July 1996.

[4] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[5] E. Dahlhaus, M. Warmuth. Membership for growing context-sensitive grammars is polynomial. *J. Comput. Sys. Sci.*, 33:456–472, 1986.

[6] S. Ginsburg, S. Greibach, J.E. Hopcroft. *Studies in Abstract Families of Languages.* Memoirs of the AMS, vol. 87. AMS, 1969.

[7] J.E. Hopcroft, J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, M.A., 1979.

[8] P. Jančar, F. Mráz, M. Plátek, J. Vogel. Restarting automata. In H. Reichel, ed., *Fundamentals of Computation Theory, Proc. FCT'95*, Lect. Notes Comput. Sci. 965, pp. 283–292. Springer, Berlin, 1995.

[9] P. Jančar, F. Mráz, M. Plátek, J. Vogel. On restarting automata with rewriting. In G. Păun, A. Salomaa, eds., *New Trends in Formal Languages*, Lect. Notes Comput. Sci. 1218, pp. 119–136. Springer, Berlin, 1997.

[10] P. Jančar, F. Mráz, M. Plátek, J. Vogel. On monotonic automata with a restart operation. *J. Autom. Lang. Comb.*, 4:287–311, 1999.

[11] T. Jurdzinski, K. Loryś, G. Niemann, F. Otto. *Some results on RRW- and RRWW-automata and their relationship to the class of growing context-sensitive languages.* Mathematische Schriften Kassel 14/01, Universität Kassel, December 2001.

[12] R. McNaughton. An insertion into the Chomsky hierarchy? In J. Karhumäki, H. Maurer, G. Păun, G. Rozenberg, eds., *Jewels are Forever*, pp. 204–212. Springer, Berlin, 1999.

[13] R. McNaughton, P. Narendran, F. Otto. Church-Rosser Thue systems and formal languages. *J. ACM*, 35:324–344, 1988.

[14] P. Narendran. *Church-Rosser and Related Thue Systems.* PhD dissertation, Rensselaer Polytechnic Institute, Troy, New York, 1984.

[15] G. Niemann, F. Otto. The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. In M. Nivat, ed., *Proc. FoSSaCS'98*, Lect. Notes Comput. Sci. 1378, pp. 243–257. Springer, Berlin, 1998.

[16] G. Niemann, F. Otto. Restarting automata, Church-Rosser languages, and representations of r.e. languages. In G. Rozenberg, W. Thomas, eds., *Proc. DLT 1999*, pp. 103–114. World Scientific, Singapore, 2000.

[17] G. Niemann, F. Otto. On the power of RRWW-automata. In M. Ito, G. Păun, S. Yu, eds., *Words, Semigroups, and Transductions*, pp. 341–355. World Scientific, Singapore, 2001.

[18] G. Niemann, F. Otto. Further results on restarting automata. In M. Ito, T. Imaoka, eds., *Words, Languages, and Combinatorics III.* World Scientific, Singapore, to appear.

[19] F. Otto, E. Moriya. *Shrinking alternating two-pushdown automata.* Mathematische Schriften Kassel 12/01, Universität Kassel, August 2001.

[20] R. Reischuk. *Komplexitätstheorie.* Teubner, Stuttgart, 1999.